IN THE SPECIFICATION

Please amend the specification as follows.

Please replace paragraph 3 on page 10 with the following paragraph:

At block 302, a set of source code modules is compiled to produce a set of object modules. For example, the compiler unit 204 compiles one or more source code modules to produce one or more object modules, which can be used as part of an application program. In one embodiment, in compiling the set of object modules, the compiler unit 204 fetches a source code module (e.g., source code module 210A) from the storage unit 202. Based on the source code module, the compiler unit 204 produces an executable object module (e.g., executable object module 212[[A]]). The compiler unit 204 can repeatedly fetch source code modules and produce object modules any number of times, according to embodiments of the invention. In one embodiment, the compiler unit produces object modules in which internal code-to-data offsets are resolved. For example, the compiler determines address distances between code and data, so that data references are relative to the beginning of a code segment or relative to the beginning of the object module. Additionally, according to embodiments, the compiler unit 204 properly page-aligned code and data segments. That is, the compiler unit 204 produces code and data segments that do not span across page boundaries when stored in a virtual memory system. The flow continues at block 303.

Please replace paragraph 2 on page 16 with the following paragraph:

As shown in Figure 6, the load commands 604 include segment commands (shown as segment command 1 and segment command 2), which specify the layout and linkage characteristics of the object module 600. According to embodiments of the invention, the load commands can include any number of segments. The load commands can include information such as the initial layout of the object module in virtual memory, and the location of a symbol table, which can be used by the loader unit 206 when resolving symbol references. The load commands can also include the initial execution state of the main program thread and the names of shared libraries that contain definitions for the executable's imported symbol. According to embodiments, the loader unit 206 can use the shared library names for resolving external references. As shown in Figure 6, the data includes two segments <u>610</u> [[310]] (shown as segment 1 and segment 2). As shown in Figure 6, segment 1 includes three sections. In particular, segment 1 includes section 1 data, section 2 data, and section 3 data. Segment 2 includes four sections. In particular, segment 2 includes section 4 data, section 5 data, section 6 data, and section 7 data. Although segments 1 and 2 include those sections shown in Figure 6, the segments can contain zero to any number of sections. Each segment defines a region of virtual memory that the loader unit can use to map into the address space of the application program. In one embodiment, the load commands and file type specify the exact number and layout of the segments and sections.

Please replace the first full paragraph at page 9 with the following paragraph:

It should be understood that the functional units (e.g., the compiler unit 204, loader unit 206, etc.) of the system 200 can be integrated or divided, forming a lesser or greater number of functional units. Moreover, the functional units can be communicatively coupled using any suitable communication method (e.g., message passing, parameter passing, and/or signals through one or more communication paths etc.). Additionally, the functional units can be connected according to any suitable interconnection architecture (fully connected, hypercube, etc.). Any of the functional units used in conjunction with embodiments of the invention can include machine readable media including instructions for performing operations described herein. Machine-readable media include any mechanism that provides (i.e., stores and/or transmits) information in a form readable by a machine (e.g., a computer). For example, a machine-readable <u>storage</u> medium includes read only memory (ROM), random access memory (RAM), magnetic disk storage media, optical storage media, <u>and</u> flash memory devices<u>.</u>[[,]] <u>Examples of a machine-readable propagation medium include</u> electrical, optical, acoustical or other forms of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.), etc. According to embodiments of the invention, the functional units can be other types of logic (e.g., digital logic) for executing the operations described herein.